Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science 149 (2006) 3–15
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Methods for Quantitative Analysis of Probabilistic Pushdown Automata

Antonín Kučera[1,2]

Faculty of Informatics, Masaryk University in Brno
Botanická 68a, 60200 Brno, Czech Republic

Abstract

We present several constructions and techniques which have recently been used to tackle the problems of qualitative/quantitative analysis of probabilistic pushdown automata.

Keywords: Probabilistic systems. Probabilistic temporal logics. Pushdown automata.

1 Introduction

Formal verification of sequential programs with recursion is a popular research subject which attracted a lot of attention in the last decade. In the non-probabilistic setting, the literature even offers two natural models for such programs:

- *pushdown automata (PDA)*, see e.g. [9,12,17,3], where the stack symbols correspond to individual procedures and their local data, and the global data is modeled in the finite-state control;
- *recursive state machines (RSM)*, see e.g. [2,1], where the behavior of each procedure is specified by a finite-state automaton which can possibly invoke the computation of another automaton in a recursive fashion.

[1] Supported by the research centre Institute for Theoretical Computer Science (ITI), project No. 1M0545.
[2] Email: kucera@fi.muni.cz

doi:10.1016/j.entcs.2005.11.013

Since PDA and RSM are fully equivalent (in a well-defined sense) and there are linear-time translations between them, the results achieved for one model immediately apply to the other. A practical impact of these results can be documented by successful applications of software tools [4,5].

Formal models for probabilistic recursive programs are obtained as probabilistic variants of PDA and RSM. The underlying semantics is given in terms of infinite-state Markov chains, and the two models are again equivalent with respect to this semantics. The existing results are described in greater detail in the following paragraphs.

Reachability.

The reachability problem is one of the most fundamental questions in formal verification. In the probabilistic setting, the problem is specified as follows: given two pPDA configurations s, t, what is the probability of reaching t from s? In particular,

- is this probability equal to one? (the qualitative reachability problem);
- is this probability bounded by a given constant? (the quantitative reachability problem).

The reachability problem was first examined in [10]. In fact, a more general result about random walks on pPDA graphs is provided; an instance of this problem is a starting configuration s and two regular sets of configurations C_1, C_2. The question is what is the probability of reaching a configuration of C_2 from s by going only through the configurations of C_1. In [10], both qualitative and quantitative variants of this problem are shown decidable.

The complexity and other algorithmic aspects of the reachability problem for probabilistic RSM were studied in greater detail in [15]. In particular, it was shown that the qualitative reachability problem for one-exit probabilistic RSM is in **P**.

LTL model-checking.

Given a pPDA configuration s and an LTL formula φ, we ask what is the probability that a run initiated in s satisfies the formula φ. Similarly as above, one can formulate the qualitative/quantitative variant of the corresponding decision problem. Moreover, one can reformulate the problem for general ω-regular properties.

In [10], it was shown that both quantitative and qualitative model-checking problem for deterministic Büchi specifications is decidable. This study was continued in [7], where the result about deterministic Büchi automata was ex-

tended to deterministic Müller automata and hence to all ω-regular properties (an elementary upper complexity bound was also given). The complexity of the model-checking problem for probabilistic RSM and ω-regular properties was studied in [13,14]. In particular, it was shown that the qualitative variant of this problem is **EXPTIME**-complete (as we already noted, these results immediately apply also to pPDA).

PCTL and PCTL* model-checking.

The model-checking problem for pPDA and branching-time probabilistic logics such as PCTL and PCTL* was studied already in [10], where it was shown that the qualitative fragment of PCTL is decidable for pPDA. In [7], it was shown that the model checking problem for pPDA and general PCTL formulae is already undecidable, while model-checking the qualitative fragment of the logic PECTL* is still decidable.

Expectations and variances.

In [11], it was shown how to compute the expected accumulated reward between two given configurations, and how to compute the corresponding variance. In particular, these results can be used to compute the expected termination time (and its variance). Moreover, it was shown how to compute the average reward per transition for infinite paths.

Long-run properties.

A long-run property is a property defined for each run w separately by considering longer and longer prefixes of w and taking the limit of the corresponding sequence of approximations. A typical example is the average service time—if a system repeatedly services certain requests, then each run can be seen as an infinite sequence of finite services. The average service time (i.e., the average number of transitions needed to service a request) for a given run w is then defined by taking the limit of averages computed from longer and longer prefixes of w. Thus, one can formulate questions like "what is the probability that the average service time for a run initiated in a given configuration s does not exceed twenty transitions?" In [6], it was shown that a rich class of long-run properties is decidable for pPDA.

****

Interestingly, most of the above given problems turned out to be decidable for pPDA, and the complexity bounds are relatively reasonable (some of these

problems are even solvable in polynomial time). In this paper we present selected techniques and constructions which have been found useful when dealing with these problems.

2 Definitions

In the paper we use $\mathbb{R}$ and $\mathbb{R}^+$ to denote the sets of real numbers and non-negative real numbers, respectively. We also use $\mathbb{R}_{\pm\infty}$ to denote $\mathbb{R} \cup \{-\infty, \infty\}$, and $\mathbb{R}^+_\infty$ to denote $\mathbb{R}^+ \cup \{\infty\}$. The symbols $-\infty, \infty$ are treated according to the standard conventions.

The underlying semantics of probabilistic sequential systems is defined in terms of discrete Markov chains.

Definition 2.1 A (discrete) *Markov chain* is a triple $M = (S, \to, \mathit{Prob})$ where S is a finite or countably infinite set of *states*, $\to \subseteq S \times S$ is a *transition relation*, and Prob is a function which to each transition $s \to t$ of M assigns its probability $\mathit{Prob}(s \to t) \in (0, 1]$ so that for every $s \in S$ we have $\sum_{s \to t} \mathit{Prob}(s \to t) = 1$.

In the rest of this paper we also write $s \xrightarrow{x} t$ instead of $\mathit{Prob}(s \to t) = x$. A *path* in M is a finite or infinite sequence $w = s_0, s_1, \cdots$ of states such that $s_i \to s_{i+1}$ for every i. The *length* of a given path w is the number of transitions in w. In particular, the length of an infinite path is ∞, and the length of a path s, where $s \in S$, is zero. We also use $w(i)$ to denote the state s_i of w (by writing $w(i) = s$ we implicitly impose the condition that the length of w is at least i). The prefix $s_0, \ldots, s_i$ of w is denoted by w^i. A *run* is an infinite path. The sets of all finite paths and all runs of M are denoted FPath and Run, respectively. Similarly, the sets of all finite paths and runs that start with a given $w \in \mathit{FPath}$ are denoted $\mathit{FPath}(w)$ and $\mathit{Run}(w)$, respectively. In particular, $\mathit{Run}(s)$, where $s \in S$, is the set of all runs initiated in s.

In this paper we are interested in probabilities of certain events that are associated with runs. To every $s \in S$ we associate the probabilistic space $(\mathit{Run}(s), \mathcal{F}, \mathcal{P})$ where $\mathcal{F}$ is the σ-field generated by all *basic cylinders* $\mathit{Run}(w)$ where $w \in \mathit{FPath}(s)$, and $\mathcal{P} : \mathcal{F} \to [0, 1]$ is the unique probability function such that $\mathcal{P}(\mathit{Run}(w)) = \Pi_{i=0}^{m-1} x_i$ where $w = s_0, \cdots, s_m$ and $s_i \xrightarrow{x_i} s_{i+1}$ for every $0 \leq i < m$ (if $m = 0$, we put $\mathcal{P}(\mathit{Run}(w)) = 1$).

Definition 2.2 A *probabilistic PDA (pPDA)* is a tuple $\Delta = (Q, \Gamma, \delta, \mathit{Prob})$ where Q is a finite set of *control states*, Γ is a finite *stack alphabet*, $\delta \subseteq Q \times \Gamma \times Q \times \Gamma^*$ is a finite *transition relation*, and Prob is a function which to each transition $pX \to q\alpha$ assigns a rational probability $\mathit{Prob}(pX \to q\alpha) \in (0, 1]$ so that for all $p \in Q$ and $X \in \Gamma$ we have that $\sum_{pX \to q\alpha} \mathit{Prob}(pX \to q\alpha) = 1$.

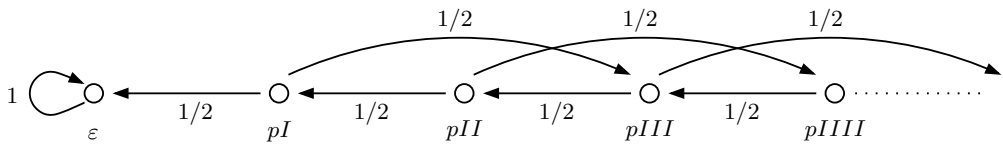

Fig. 1. The Markov chain M_Δ.

In the rest of this paper we adopt a more intuitive notation, writing $pX \to q\alpha$ instead of $(p, X, q, \alpha) \in \delta$, and $pX \xrightarrow{x} q\alpha$ instead of $Prob(pX \to q\alpha) = x$. The set $Q \times \Gamma^*$ of all configurations of Δ is denoted by $\mathcal{C}(\Delta)$. Given a configuration $pX\alpha$, we call pX the *head* and α the *tail* of $pX\alpha$.

To Δ we associate the Markov chain M_Δ where $\mathcal{C}(\Delta)$ is the set of states and the transitions are determined as follows:

- $p\varepsilon \xrightarrow{1} p\varepsilon$ for each $p \in Q$ (here ε denotes the empty stack);
- $pX\beta \xrightarrow{x} q\alpha\beta$ is a transition of M_Δ iff $pX \xrightarrow{x} q\alpha$ is a transition of Δ.

Example 2.3 As a simple example, we can take a pPDA Δ with one control state p, one stack symbol I, and two transitions $pI \xrightarrow{1/2} p\varepsilon$, $pI \xrightarrow{1/2} pIII$. The (infinite-state) Markov chain M_Δ is shown in Fig. 1.

3 Recursive equations

In this section we show that various quantities and numerical features of pPDA which depend only on finite prefixes of runs can be obtained as the least solution of an effectively constructible system of mutually recursive equations. This approach has been used in [10,15] to solve the quantitative/qualitative reachability problem, and later also in [11] to handle the expected values and variances of certain random variables defined over runs in pPDA. Since these equations typically contain just summation and multiplication, they can be used to express the associated properties in $(\mathbb{R}, +, *, \leq)$, i.e., first-order theory of the reals. As $(\mathbb{R}, +, *, \leq)$ is decidable [16] and some fragments even have a relatively reasonable complexity [8], one can thus obtain upper complexity bounds for some of the considered problems.

For the rest of this section, we fix a pPDA $\Delta = (Q, \Gamma, \delta, Prob)$. We can safely assume that each transition in δ is of the form $rY \to t\alpha$, where the length of α is at most 2. This is no restriction, because each pPDA can efficiently be transformed so that this condition is satisfied and the underlying Markov chain is the same upto renaming the states.

3.1 Reachability

Let us first examine the reachability problem in greater detail. Let pX a configuration of Δ, and $q \in Q$. We are interested in the probability that a run initiated in pX visits the configuration $q\varepsilon$. Let us denote this probability by $[pXq]$. Formally, $[pXq]$ is defined as follows (cf. the definition of the probabilistic space $(Run(s), \mathcal{F}, \mathcal{P})$ in Section 2):

$$[pXq] = \mathcal{P}(\{w \in Run(pX) \mid w(i) = q\varepsilon \text{ for some } i\})$$

This means that $[pXq]$ can be expressed as the following infinite sum, where $FPath(pX, q\varepsilon)$ denotes the set of all finite paths initiated in pX that terminate in $q\varepsilon$ (note that $q\varepsilon$ cannot be revisited along such a path):

$$[pXq] = \sum_{w \in FPath(pX, q\varepsilon)} \mathcal{P}(Run(w))$$

So, one possible strategy for computing $[pXq]$ is to manipulate the above sum until it reaches a closed form. If $FPath(pX, q\varepsilon)$ has a simple and regular structure, this might be possible. However, for general pPDA this approach is not very useful. Fortunately, there is a simpler way of "computing" $[pXq]$ which utilizes the mutual recursive dependency among the probabilities of the finite family $\{[sYt] \mid s, t \in Q, Y \in \Gamma\}$. For example, let us consider the simple pPDA of Example 2.3. It is not hard to see that $[pIp]$ has to satisfy the following equation:

$$[pIp] \quad = \quad \frac{1}{2} \; + \; \frac{1}{2} \cdot [pIp] \cdot [pIp] \cdot [pIp] \tag{1}$$

Intuitively, this is because *every* path from pI either hits $p\varepsilon$ immediately (this happens with probability 1/2), or goes to $pIII$ (which also happens with probability 1/2). Now, every path from $pIII$ to $p\varepsilon$ can be split into three parts as follows:

$$pIII \longrightarrow^* pII \longrightarrow^* pI \longrightarrow^* p\varepsilon$$

The "splitting points" are the *first* occurrences of pII and pI along a given path. Note that since the stack length can be decreased at most by one in a single transition, every path from $pIII$ to $p\varepsilon$ has to go through pII and pI. A closer look at subpaths from $pIII$ to pII reveals that the total probability of all these subpaths is the same as the total probability of all finite path from pI to $p\varepsilon$ (here we use the fact that the stack length in all of the intermediate configurations between $pIII$ and pII is at least 3). A similar observation holds also for the subpaths from pII to pI. Thus, we obtain the factor $[pIp] \cdot [pIp] \cdot [pIp]$.

Equation (1) has the following three solutions: $-\frac{\sqrt{5}+1}{2}$, 1, $\frac{\sqrt{5}-1}{2}$. The first solution is not in $[0, 1]$ and can be disregarded immediately. The other two

solutions are eligible, and one can show (by investing some effort) that the *least* eligible solution is the right one. That is, $[pIp] = \frac{\sqrt{5}-1}{2}$ (the "golden ratio").

The technique used in the previous example can be extended to general pPDA as follows. For all $r, s \in Q$ and Y we introduce a fresh variable $R(rYs)$ and its corresponding equation

$$R(rYs) = \sum_{rY \xrightarrow{x} s\varepsilon} x + \sum_{rY \xrightarrow{x} tU} x \cdot R(tUs) + \sum_{rY \xrightarrow{x} tVZ} x \cdot \sum_{c \in Q} R(tVc) \cdot R(cZs) \tag{2}$$

Thus, we obtain a finite system of recursive equations whose size is polynomial in the size of Δ.

Theorem 3.1 (see [10]) *The tuple of all $[rYs]$ probabilities is exactly the tuple of values from $[0,1]$ that forms the* least *solution (wrt. component-wise ordering) of the system of recursive equations constructed above.*

In general, the least solution of the above system of equations cannot be given analytically (in the previous example, we managed to do that because here the system contains just one simple equation for $[pIp]$). Moreover, these probabilities can take irrational values (note that $[pIp]$ is irrational) and therefore cannot be computed precisely. Nevertheless, for each $[pXq]$ we can effectively construct a formula Φ of $(\mathbb{R}, +, *, \leq)$ with one free variable z such that $\Phi(z/c)$ holds iff $c = [pXq]$. Intuitively, the formula Φ says

> "there is a tuple $\vec{R}$ of values from $(0,1)$ which forms the least solution of the system of equations constructed above, and z is equal to the element of $\vec{R}$ which corresponds to $[pXq]$."

Since $(\mathbb{R}, +, *, \leq)$ is decidable, the qualitative reachability problem is also decidable—we simply ask whether $\Phi(z/1)$ holds or not. Similarly, the quantitative reachability problem can be solved by deciding the validity of the formula $\exists z : \Phi \wedge z \leq c$.

3.2 The expected termination time

The approach of Section 3.1 can also be used to express other interesting features of pPDA. For example, one can ask what is the expected number of transitions needed to reach $q\varepsilon$ from pX. Formally, for all $p, q \in Q$ and $X \in \Gamma$ we define a random variable $T[pXq] : Run(pX) \to \mathbb{R}^+_\infty$ as follows:

$$T[pXq](w) = \begin{cases} n & \text{if } w \text{ hits } q\varepsilon \text{ after exactly } n \text{ transitions} \\ \infty & \text{if } w \text{ does not hit } q\varepsilon \end{cases}$$

Note that if $[pXq] < 1$, then the expected value of $T[pXq]$, denoted $E(T[pXq])$, is equal to ∞. In this case, it is more reasonable to ask what is *conditional* ex-

pected value of $T[pXq]$ under the condition that $q\varepsilon$ is indeed reached from pX. Let us denote this conditional expectation $E[pXq]$. Note that $E[pXq]$ is defined only if $[pXq] > 0$; also note that if $[pXq] = 1$, then $E[pXq] = E(T[pXq])$. It can also happen that $E[pXq] = \infty$ even if $[pXq] = 1$.

The value of $E[pXq]$ can be expressed in a similar way as the value of $[pXq]$. For all $r, s \in Q$ and Y we introduce a fresh variable $V(rYs)$ and its corresponding equation

$$V(rYs) \;=\; \frac{1}{[rYs]} \cdot \Bigg(\sum_{rY \xrightarrow{x} s\varepsilon} x \;+\; \sum_{rY \xrightarrow{x} tU} x \cdot [tUs] \cdot (1 + V(tUs)) \;+\; \sum_{rY \xrightarrow{x} tVZ} x \cdot \sum_{c \in Q} [tVc] \cdot [cZs] \cdot (1 + V(tVc) + V(cZs)) \Bigg)$$

Again, it can be shown that the tuple of all $E(rYs)$ forms the least solution of the constructed system of equations (now the components take values from $\mathbb{R}^+_\infty$). Actually, we setup these equations only for those $V(rYs)$ where $[rYs] > 0$, and delete all summands which contain "undefined" variables. Note that these equations are *linear* because the probabilities of the form $[tUs]$ are now interpreted as constants. The intuition behind these equations is similar as before. The runs from rY to $s\varepsilon$ can be split into disjoint subsets according to the first transition performed. Let us consider, e.g., those runs which start with the transition $rY \xrightarrow{x} tVZ$. If we are to reach $s\varepsilon$ eventually, we have to go through some configuration of the form cZ for the first time. As soon as we fix this cZ, we can evaluate the probability of all such runs—obviously, this probability is equal to $x \cdot [tVc] \cdot [cZs]$. Now we need to express the associated number of transitions. There is the initial transition $rY \xrightarrow{x} tVZ$. Then we need to go from tVZ to cZ (so that the stack length never drops to 1 in between). Of course, this number of transitions can be different in every run, but in average it is equal to $E(tVc)$. Hence, we use $V(tVc)$ to denote this value. Similarly, the number of transitions from cZ to $s\varepsilon$ is $E(cZs)$ in average, and we use $V(cZs)$ to denote the corresponding value. Thus, we obtain the sum $1 + V(tVc) + V(cZs)$.

The previous explanation is only intuitive and does not rely on rigorous arguments. A formal proof that the constructed equations indeed "behave" in the indicated way is not completely trivial. In particular, one has to use the strong law of large numbers to justify the use of the $V(\ldots)$ variables in the right-hand sides of these equations. Details can be found in [11].

The constructed system of equations allows to express the conditional expectations $E(pXq)$ in $(\mathbb{R}, +, *, \leq)$ along the same lines as in Section 3.1. However, there are some technical subtleties. For example, some of these expecta-

tions can be infinite and this requires some care. Another problem is that the probabilities of the form $[tVc]$ appear as "known constants" in the constructed linear equations. However, these can be handled fully symbolically. For each such probability $[tVc]$ we simply fix a fresh variable $Y(tVc)$ which is used in these linear equations instead of $[tVc]$, and bind the value of $Y(tVc)$ to $[tVc]$ using the corresponding formula Φ (see Section 3.1).

3.3 Further remarks

In the previous two subsections we indicated how to express certain numerical features of pPDA by systems of recursive equations. This allows to represent these numbers symbolically by certain formulae of $(\mathbb{R}, +, *, \leq)$. Note that once a certain number is shown to be effectively expressible in $(\mathbb{R}, +, *, \leq)$, it can be treated almost as a constant—in particular, it can be freely used in other formulae which define other (possible more complicated) numerical features of pPDA. We have seen this in Section 3.2, where the probabilities of the form $[tVc]$ were used as constants in equations which define the conditional expectations $E(pXq)$. One can go on in this direction—for example, in [11] it was shown how to express the conditional variance of the number of transitions needed to reach $q\varepsilon$ from pX, under the condition that $q\varepsilon$ is reached from pX. In the corresponding equations, both the probabilities $[tVc]$ and the conditional expectations $E(tYc)$ are used as known constants.

Finally, let us note that the presented results can be generalized to certain classes of *reward functions* which assign to each configuration of Δ a non-negative reward. Then one can ask, e.g., what is the conditional expected reward accumulated along a path from pX to $q\varepsilon$, under the condition that $q\varepsilon$ is reached from pX. We refer to [11] for details.

4 The Markov chain X_Δ

The method presented in Section 3 is useful when dealing with those properties of pPDA which are determined only by finite prefixes of runs. However, this method does not allow to handle properties that depend on "complete" runs. For example, the (in)validity of an LTL formula for a given run w cannot be deduced just from some (sufficiently long) finite prefix of w. Hence, another approach is needed to tackle the problem of qualitative/quantitative LTL model-checking, as well as other problems about limit properties of runs.

In this section we present a generic construction which has been used to compute various numerical features of pPDA that take into account limit properties of runs. Roughly speaking, we show how to represent the behaviour of a given pPDA Δ by a *finite* Markov chain X_Δ which preserves the property

of our interest. The idea behind the construction of X_Δ is not so easy to explain at an intuitive level. Therefore, we select just one (simple) problem and show how it can be solved. Then, we give some hints on how one can apply this method to more complicated problems.

Let $M = (S, \to, Prob)$ be a Markov chain, and let $F \subseteq S$ be a finite subset of *final* states. A run w is *recurring* if it visits a final state infinitely often. Our aim is to compute the probability $\mathcal{P}(s, Inf)$ of all recurring runs from a given $s \in S$.

4.1 Computing $\mathcal{P}(s, Inf)$ for finite-state Markov chains

For simplicity, we first show how to compute $\mathcal{P}(s, Inf)$ for finite-state Markov chains. So, let us fix a finite-state Markov chain $M = (S, \to, Prob)$, a state s of S, and a subset $F \subseteq S$ of final states. The first step is to decompose the (graph of) M into strongly connected components (SCCs). Since M is finite, there must be some *final* SCCs from which there are no outgoing transitions. For example, the chain of Fig. 2 has two final SCCs denoted C_1 and C_2. Let C be a final SCC and let $Run(s, C)$ be the set of all runs initiated in s that eventually hit the component C. Let $C_1, \cdots, C_n$ be the final SCCs of M. A standard result of Markov chain theory is that

$$\sum_{i=1}^{n} \mathcal{P}(Run(s, C_i)) = 1 \tag{3}$$

That is, almost all runs eventually hit a final SCC. Since M is finite, each $\mathcal{P}(Run(s, C_i))$ is computable by standard methods. Another standard result (ergodic theorem) implies that almost all runs of $Run(s, C_i)$ visit each state of C_i infinitely often. Hence, if C_i contains at least one final state, then almost all runs of $Run(s, C_i)$ are recurring. Otherwise, no run of $Run(s, C_i)$ is recurring. Let $\mathcal{C}$ be the set of all final SCCs which contain at least one final state. Then

$$\mathcal{P}(s, Inf) = \sum_{C \in \mathcal{C}} \mathcal{P}(Run(s, C)) \tag{4}$$

Hence, $\mathcal{P}(Run(s, Inf))$ is computable.

4.2 Computing $\mathcal{P}(s, Inf)$ for pPDA

Now we show how to compute $\mathcal{P}(s, Inf)$ for pPDA. For the rest of this section, we fix a pPDA $\Delta = (Q, \Gamma, \delta, Prob)$ and a subset $F \subseteq Q$ of control states. A configuration $p\alpha$ of Δ is *final* if $p \in F$. We also fix an initial configuration $q_0 Z_0$, and we assume that Z_0 cannot be removed from the stack, i.e., $q_0 Z_0 \not\to^* q\varepsilon$ for any $q \in Q$. Our aim is to compute the probability $\mathcal{P}(q_0 Z_0, Inf)$.

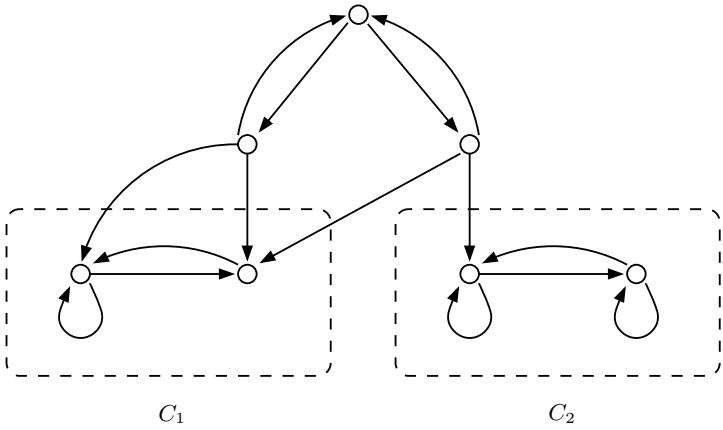

Fig. 2. The chain M (transition probabilities are not shown).

Since the chain M_Δ is infinite, we cannot use the approach of Section 4.1 directly. The crucial step is the construction of another finite-state Markov chain X_Δ which is described next.

Let $w \in Run(q_0 Z_0)$. A configuration $p_i\alpha_i$ of w, where $i \geq 0$, is *minimal* if $|\alpha_i| \leq |\alpha_j|$ for all $j > i$. The *k-th minimum* of w, denoted $\min_k(w)$, is the k-th minimal configuration of w. Intuitively, the minimal configurations of a given run are exactly the positions where one can forget about the stack content below the top-of-the-stack symbol, because these symbols are never accessed in the future. This intuition is formally captured in our next definitions.

For every $i \in \mathbb{N}$ we define a random variable X_i over $Run(q_0 Z_0)$ as follows: $X_i(w) = (qY, m)$, where qY is the head of $\min_i(w)$, and m is either 0 or 1 depending on whether a final configuration was visited between $\min_{i-1}(w)$ and $\min_i(w)$ (here the configuration $\min_{i-1}(w)$ does *not* count, while $\min_i(w)$ does). In particular, $X_1(w)$ is equal either to $(q_0 Z_0, 1)$ or to $(q_0 Z_0, 0)$, depending on whether $q_0 \in F$ or not, respectively. A technical proof (see, e.g., [10]) reveals that the sequence of random variables $X_\Delta = X_1, X_2, \cdots$ is a Markov chain. To get some intuition why, let us examine the probability

$$\mathcal{P}(X_k{=}(q_k Y_k, m_k) \mid X_{k-1}{=}(q_{k-1} Y_{k-1}, m_{k-1}) \wedge \cdots \wedge X_1{=}(q_1 Y_1, m_1))$$

where $\mathcal{P}(\bigwedge_{i=1}^{k-1} X_i{=}(q_i Y_i, m_i)) > 0$. Intuitively, the values of $X_1, \cdots, X_{k-2}$ do not really matter, because the symbols which are stored below top-of-the-stack symbol at $\min_{k-1}(w)$ are not accessed anyway. Hence, it is not important what was the sequence of previous minimal configurations. Note that X_Δ has $2 \cdot |Q| \cdot |\Gamma|$ states.

The transition probabilities in X_Δ can take irrational values, but are effectively expressible in $(\mathbb{R}, +, *, \leq)$ (cf. Section 3.1). Now we can use the same argument as in Section 4.1, i.e., we identify the final SCCs of X_Δ, and "com-

pute" (i.e., express in $(\mathbb{R}, +, *, \leq)$) the probability of reaching a final SCC which contains at least one state of the form $(qY, 1)$.

4.3 Further remarks

The main idea behind the construction of X_Δ is to abstract each run w of Δ by a sequence of heads of the minimal configurations of w. Since this abstraction does not faithfully reflect the subpaths between two consecutive minimal configurations, some auxiliary information must be added to X_Δ in order to capture the property of interest. This can be done in various ways according to specific needs of a given property. For example, in [10,7] this approach was used to compute (i.e., express in $(\mathbb{R}, +, *, \leq)$) the probability of all runs that satisfy a given ω-regular property. In [11], it was shown how to compute the expected *gain*, i.e., the average reward per transition for pPDA. This study was continued in [6], where the chain X_Δ was used to compute a large class of long-run average properties of pPDA.

References

[1] R. Alur, S. Chaudhuri, K. Etessami, and P. Madhusudan. On-the-fly reachability and cycle detection for recursive state machines. In *Proceedings of TACAS 2005*, vol. 3440 of *Lecture Notes in Computer Science*, pp. 61–76. Springer, 2005.

[2] R. Alur, K. Etessami, and M. Yannakakis. Analysis of recursive state machines. In *Proceedings of CAV 2001*, vol. 2102 of *Lecture Notes in Computer Science*, pp. 207–220. Springer, 2001.

[3] R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proceedings of STOC 2004*, pp. 202–211. ACM Press, 2004.

[4] T. Ball and S.K. Rajamani. Bebop: A symbolic model checker for boolean programs. In *SPIN 00: SPIN Workshop*, vol. 1885 of *Lecture Notes in Computer Science*, pp. 113–130. Springer, 2000.

[5] T. Ball and S.K. Rajamani. The SLAM project: debugging system software via static analysis. In *Proceedings of POPL 2002*, pp. 1–3. ACM Press, 2002.

[6] T. Brázdil, J. Esparza, and A. Kučera. Analysis and prediction of the long-run behavior of probabilistic sequential programs with recursion. In *Proceedings of FOCS 2005*, pp. 521–530. IEEE Computer Society Press, 2005.

[7] T. Brázdil, A. Kučera, and O. Stražovský. On the decidability of temporal properties of probabilistic pushdown automata. In *Proceedings of STACS'2005*, vol. 3404 of *Lecture Notes in Computer Science*, pp. 145–157. Springer, 2005.

[8] J. Canny. Some algebraic and geometric computations in PSPACE. In *Proceedings of STOC'88*, pp. 460–467. ACM Press, 1988.

[9] J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In *Proceedings of CAV 2000*, vol. 1855 of *Lecture Notes in Computer Science*, pp. 232–247. Springer, 2000.

[10] J. Esparza, A. Kučera, and R. Mayr. Model-checking probabilistic pushdown automata. In *Proceedings of LICS 2004*, pp. 12–21. IEEE Computer Society Press, 2004.

[11] J. Esparza, A. Kučera, and R. Mayr. Quantitative analysis of probabilistic pushdown automata: Expectations and variances. In *Proceedings of LICS 2005*, pp. 117–126. IEEE Computer Society Press, 2005.

[12] J. Esparza, A. Kučera, and S. Schwoon. Model-checking LTL with regular valuations for pushdown systems. *Information and Computation*, 186(2):355–376, 2003.

[13] K. Etessami and M. Yannakakis. Algorithmic verification of recursive probabilistic systems. In *Proceedings of TACAS 2005*, vol. 3440 of *Lecture Notes in Computer Science*, pp. 253–270. Springer, 2005.

[14] K. Etessami and M. Yannakakis. Checking LTL properties of recursive Markov chains. In *Proceedings of 2nd Int. Conf. on Quantitative Evaluation of Systems (QEST'05)*, 2005.

[15] K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of non-linear equations. In *Proceedings of STACS'2005*, vol. 3404 of *Lecture Notes in Computer Science*, pp. 340–352. Springer, 2005.

[16] A. Tarski. *A Decision Method for Elementary Algebra and Geometry.* Univ. of California Press, Berkeley, 1951.

[17] I. Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164(2):234–263, 2001.